Project Report

Project Title: Checkmate Revolution: A Four-Player Chess Variant with Rotating Boards

Submitted By:

Aazar Arnold (22K-4277), Bilal Ansari (22K-4365), Umer Nadeem (22K-4158)

Course: AI

Instructor: Ms. Alishba Subhani (Lab), Ms. Ramsha Jatt (Theory)

Submission Date: 11/5/2025

1. Executive Summary

This project aimed to reimagine traditional chess into a dynamic four-player format featuring a rotating board and AI-controlled opponents. Titled Checkmate Revolution, the game involved extensive rule modifications, visual updates, and custom AI logic using the Minimax algorithm enhanced with Alpha-Beta Pruning. The AI was developed to strategically play against human or other AI players by evaluating multiple board states and choosing optimal moves. Over 20 iterative versions were built to reach the final stable release.

2. Introduction

Chess is a classic two-player strategy game dating back centuries. In this project, we reimagined chess to support four simultaneous players with turn-based logic and board rotation every few turns. The innovation also included AI opponents, allowing solo or mixed matches. This version promotes multiplayer engagement and increased complexity, demanding deeper strategy.

Objectives:

- Implement a functional four-player chess game with rotating board mechanics.

- Integrate AI using Minimax and Alpha-Beta pruning.

- Develop responsive UI with piece visuals.

- Resolve rendering, rule logic, and performance issues iteratively.

3. Game Description

Original Game Rules:

Standard chess includes two players, each controlling 16 pieces on an 8x8 board. The objective is to checkmate the opponent's king using strategic movement and capture rules for each piece.

Innovations:

- Four-player format (Red, Blue, Green, Yellow).
- Turn-based rotation after each full round.
- Custom piece rendering logic using initials or visuals.
- AI logic for autonomous opponents.
- Checkmate detection per player.
- Simplified piece identification to distinguish Knight and King (e.g., 'N' for knight).

4. AI Approach and Methodology

AI Techniques:

We employed the Minimax algorithm, optimized with Alpha-Beta pruning, to enable the AI to analyze multiple move trees while minimizing computational overhead.

Heuristics:

A custom evaluation function scored board states by piece value, positioning, and threat detection. The AI penalizes repetitive moves using move history, considers check states, and uses a transposition table to cache board evaluations.

Performance:

AI was evaluated based on decision speed, ability to avoid illegal or non-beneficial moves, and performance in matches against human players. We introduced early escape detection for checks and move pruning to maintain fluidity.

5. Game Mechanics and Rules

Modified Game Rules:

- Four players with individual turn sequences.
- Each player plays as a different color.
- Rotation occurs after each player completes a move.
- Elimination on checkmate does not end the game.

Turn Sequence:

Player selects piece -> Valid moves highlighted -> Move executed -> Board rotates -> Next player begins.

Winning:

Game continues until only one player remains not checkmated.

6. Implementation and Development

Development Process:

The game was developed in Python using the Pygame library. Initial versions used graphical piece images, but due to rendering issues, we transitioned to simpler initials with color-coded visuals. Over time, we implemented piece logic fixes, checkmate logic, and AI performance enhancements.

Languages/Tools:

- Language: Python
- Libraries: Pygame, NumPy
- Tools: GitHub for version control

Challenges:

- Board Orientation
- Piece Conversion Logic
- Sprite and Image Handling
- Checkmate Logic
- Turn Management and AI Loops

7. Team Contributions

- Aazar Arnold (22K-4277): Lead on AI architecture and game logic, designed and optimized Minimax with Alpha-Beta pruning.
- Bilal Ansari (22K-4365): Focused on UI rendering, board rotation logic, and debugging visual representation issues.
- Umer Nadeem (22K-4158): Implemented turn-based mechanics, checkmate detection, and integration of AI with the main gameplay loop.

8. Results and Discussion

AI Performance:

After extensive tuning:

- ~75% win rate in 4-player simulations
- Average move decision: < 2 seconds
- Avoids repetition using move history
- Handles checks and prioritizes check escapes effectively

9. References

- Russell, S. & Norvig, P. (2010). Artificial Intelligence: A Modern Approach
- Pygame Documentation: https://www.pygame.org/docs/
- Chess Programming Wiki: https://www.chessprogramming.org/Main_Page
- StackOverflow (Various Pygame issues)
- GitHub community contributions (4-player chess adaptations)